METHOD AND APPARATUS FOR REDUCING CACHE THRASHING

Field of the Invention

The present invention relates generally to cache memory devices, and more particularly, to methods and apparatus for adaptively decreasing cache trashing in a cache memory device.

Background of the Invention

Processors often use a cache to improve performance and decrease system costs. Caches temporarily store recently accessed information (blocks of instructions or data) in a small memory that is faster to access than a larger main memory. Caches are effective because a block that has been accessed once is likely to be accessed soon again or is often near a recently accessed block. Thus, as a task executes, the working set of a task (the instructions and data currently required for the task) is stored in the cache in the event that the information may be accessed again. A cache typically maps multiple blocks of information from the main memory into one place in a cache, typically referred to as a "set." A "block" refers to the minimum unit of information that can be present in a cache and a "frame" is the place in a cache where a single block may be stored. In a set associative cache, multiple frames are grouped into sets. For example, as two-way set associative cache has two frames in each set.

Index bits in the address of a block of main memory select a set to hold the block in the cache. The index bits thus associate (map) a block to a cache set. Any of the frames in the set may hold a block that maps to the set. When a new block is stored in a set, it is typically stored in the least recently accessed frame of the set. A block that currently resides in the frame (if any) is evicted from the cache. A cache directory is typically used to determine if a given block is in the cache. The directory is often a table of entries, one entry for each set in the cache. An entry contains one field for each frame in its associated set. To determine if a block is in the cache, the corresponding index bits are used to select a directory entry. If a given block is in the cache, the address of the block is in one of the entry's frame fields.

The index bits specify the set that holds the block in the cache. Thus, all blocks with the same index map to the same set. If there are not enough frames in a set to store all the blocks that map to the set that are currently in use by a program, one or more frames must be evicted prematurely (i.e., before temporally local accesses to them have completed), thereby increasing cache misses. This phenomenon is referred to as "thrashing" and can significantly decrease cache performance. A need therefore exists for an adaptive mechanism to decrease cache trashing in a cache memory device. A further need exists for a mechanism for extending a cache set when such thrashing occurs.

Summary of the Invention

Generally, a method and apparatus are disclosed for adaptively decreasing cache trashing in a cache memory device. The present invention improves performance of a cache by automatically detecting thrashing of a set and then providing one or more augmentation frames as additional cache space. In one embodiment, the augmentation frames are obtained by mapping the blocks that map to a thrashed set to one or more additional, less utilized sets.

The disclosed cache thrashing reduction system initially identifies a set that is likely to be experiencing thrashing, referred to herein as a thrashed set. Once thrashing is detected, the cache thrashing reduction system selects one or more additional sets to augment a thrashed set, referred to herein as the augmentation sets. In this manner, blocks of main memory that are mapped to a thrashed set are now mapped to an expanded group of sets (the thrashed set and the augmentation sets). Finally, when the augmentation sets are no longer likely to be needed to decrease thrashing, the augmentation set(s) are disassociated from the thrashed set(s).

Thrashed set detection may be based, for example, on the individual miss rate of a set, the miss rate of a set relative to other sets, the addresses of the blocks involved in misses on the set, or a combination of the foregoing. An exemplary approach assumes that a set that is experiencing a high miss rate may be experiencing thrashing. In one implementation, a miss counter and an access counter are associated with one or more sets. The accesses of a given set are counted and the miss rate is determined by comparing the number of misses experienced during a given number of accesses. A reduction in logic may be achieved by counting accesses to a group of sets, such as set pairs, rather than to individual sets. In a further variation, only the misses and accesses relative to sets that have recently experienced a miss are counted.

Thrashing is reduced on a set in accordance with the present invention by selecting one or more additional sets (augmentation sets) in the cache to share their space with the thrashed set. The augmentation sets may be selected, for example, based on a low access rate, a position in the address space relative to a thrashed set or their miss rate, using a static assignment, a wired-in assignment or by other means. When a given set augments a thrashed set, it shares its space with the thrashed set. In a unidirectional augmentation approach, some blocks that formerly mapped to the thrashed set now map to an augmentation set and blocks that previously mapped to the augmentation set continue to do so. In a bidirectional augmentation approach, the blocks that map to either the thrashed set or to the augmentation set are distributed across both sets. That is, some blocks that previously mapped to the thrashed set now map to the augmentation set and some blocks that previously mapped to the augmentation set now map to the thrashed set. In one exemplary implementation, augmentation set(s) are selected based on a cache index of a set relative to that of a thrashed set.

There are a number of ways to map some of the blocks mapped to a thrashed set to an augmentation set (to couple the augmentation set to the thrashed set). Generally, coupling should ensure that blocks that are thrashing would be mapped in approximately equal numbers to the thrashed set and to the augmentation set. An exemplary concurrent symmetric static-pairs coupling approach is disclosed where each set in a pair is the augmentation set for the other and if thrashing is detected on either set, both sets share blocks mapped to them with the other set. In non-concurrent versions, the thrashed set shares blocks mapped to it with an augmentation set but the augmentation set would not share its blocks with the thrashed set.

According to a further aspect of the present invention, a mechanism is disclosed for releasing an augmentation set from the map of a thrashed set, when appropriate. While a larger cache may help to decrease thrashing, this entails a larger directory and applies additional space everywhere, whereas space is actually needed only in specific places for limited amounts of time. The additional area of a larger cache is apt to be greater than that consumed by the anti-thrashing logic of the present invention. The present invention effectively enhances cache performance by selectively augmenting over-utilized cache space with under-utilized space, when needed.